# The Template Update Problem

Iain Matthews, *Member*, *IEEE*, Takahiro Ishikawa, and Simon Baker

**Abstract**—Template tracking dates back to the 1981 Lucas-Kanade algorithm. One question that has received very little attention, however, is how to update the template so that it remains a good model of the tracked object. We propose a template update algorithm that avoids the "drifting" inherent in the naive algorithm.

**Index Terms**—Template tracking, the Lucas-Kanade algorithm, active appearance models.

## 1 Introduction

TEMPLATE tracking is a well studied problem in computer vision which dates back to [7]. An object is tracked through a video by extracting an example image of the object, a *template*, in the first frame and then finding the region which matches the template as closely as possible in the remaining frames. Template tracking has been extended in a variety of ways, including: 1) to allow arbitrary parametric transformations of the template [3], 2) to allow linear appearance variation [4], [6], and 3) to be efficient [6], [2]. Combining these extensions to Lucas-Kanade has resulted in the real-time fitting of nonrigid appearance models such as Active Appearance Models (AAMs) [5], [8].

The underlying assumption behind template tracking is that the appearance of the object remains the same throughout the entire video. This assumption is generally reasonable for a certain period of time, but eventually the template is no longer an accurate model of the appearance of the object. A naive solution to this problem is to update the template every frame (or every $n$ frames) with a new template extracted from the current image at the current location of the template. The problem with this naive algorithm is that the template "drifts." Each time the template is updated, small errors are introduced in the location of the template. With each update, these errors accumulate and the template steadily drifts away from the object. See Fig. 1 for an example.

In this paper, we propose a template update algorithm that does not suffer from this drift. The template can be updated in every frame and yet still stays firmly attached to the original object. The algorithm is a simple extension of the naive algorithm. As well as maintaining a current estimate of the template, our algorithm also retains the first template from the first frame. The template is first updated as in the naive algorithm with the image at the current template location. To eliminate drift, this updated template is then aligned with the first template to give the final update. We first evaluate our algorithm *qualitatively* and show that it can update the template without introducing drift. Next, we reinterpret our algorithm as a heuristic to avoid local minima. We then *quantitatively* evaluate the algorithm as a technique to avoid local minima.

Next, we consider the more general case of template tracking with linear appearance variation. Specifically, we generalize our algorithm to AAMs [5]. In this context, our appearance update algorithm can also be interpreted as a heuristic to avoid local minima and, so, we again quantitatively evaluate it as such. We also demonstrate how our algorithm can be applied to convert a generic a person-independent AAM into a person-specific AAM.

• *The authors are with the The Robotics Institute, Carnegie Mellon University, 500 Forbes Ave., Pittsburgh, PA 15232.*
*E-mail: {iainm, taka, simon}@cs.cmu.edu.*

*Manuscript received 14 Aug. 2003; revised 21 Jan. 2004; accepted 27 Feb. 2004.*
*Recommended for acceptance by J. Weng.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0230-0803.*



## 2 Single Template Tracking

We begin by considering the original template tracking problem [7] where the object is represented by a single template image. Suppose we are given a video sequence of images $I_n(\mathbf{x})$ where $\mathbf{x} = (x, y)^{\mathrm{T}}$ are the pixel coordinates and $n = 0, 1, 2, \ldots$ is the frame number. In template tracking, a subregion of the initial frame $I_0(\mathbf{x})$ that contains the object of interest is extracted and becomes the template $T(\mathbf{x})$. The template is not necessarily rectangular, and might, for example, be a face shaped region [5], [8].

Let $\mathbf{W}(\mathbf{x};\mathbf{p})$ denote the parameterized set of allowed deformations of the template, where $\mathbf{p} = (p_1, \ldots, p_k)^{\mathrm{T}}$ is a vector of parameters. The warp $\mathbf{W}(\mathbf{x};\mathbf{p})$ takes the pixel $\mathbf{x}$ in the coordinate frame of the template $T(\mathbf{x})$ and maps it to a subpixel location $\mathbf{W}(\mathbf{x};\mathbf{p})$ in the coordinate frame of the video $I_n(\mathbf{x})$. The set of allowed warps depends on the type of motions we expect from the object being tracked. If the object is a roughly planar image patch moving in 2D, we might consider the set of *similarity warps*:

$$W(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & - & p_2 \cdot y & + & p_3 \\ p_2 \cdot x & + & (1+p_1)\cdot y & + & p_4, \end{pmatrix}, \quad (1)$$

where there are four parameters $\mathbf{p} = (p_1, p_2, p_3, p_4)^{\mathrm{T}}$. In general, the number of parameters $k$ may be arbitrarily large and $\mathbf{W}(\mathbf{x};\mathbf{p})$ can be arbitrarily complex [3]. A particularly complex example is the set of piecewise affine warps used to model nonrigidly moving objects in Active Appearance Models (AAMs) [5], [8].

The goal of template tracking is to find the best match to the template in every subsequent frame in the video. The sum of squared error is normally used to measure the degree of match between the template and the video frames. The goal is therefore to compute:

$$\mathbf{p}_n = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in T} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2 \quad (2)$$

for $n \geq 1$ and where the summation is over all of the pixels in the template (a convenient abuse of terminology). The original solution to the nonlinear optimization in (2) was the Lucas-Kanade algorithm [7]. A variety of other algorithms have since been proposed. See [2] for a recent survey.

### 2.1 Template Update Strategies

In this paper, we consider the problem of how to update the template $T(\mathbf{x})$. Suppose that a (potentially) different template is used in each frame. Denote the template that is used in the $n$th frame $T_n(\mathbf{x})$. Tracking then consists of computing:

$$\mathbf{p}_n = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in T_n} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x})]^2 \quad (3)$$

and the template update problem consists of computing $T_{n+1}(\mathbf{x})$ from the images $I_0(\mathbf{x}), \ldots, I_n(\mathbf{x})$ and the templates $T_1(\mathbf{x}), \ldots, T_n(\mathbf{x})$. The simplest strategy is not to update the template at all:

**Strategy 1: No Update**

$$T_{n+1}(\mathbf{x}) = T_1(\mathbf{x}) \text{ for all } n \geq 1.$$

The simplest strategy for actually updating the template is to set the new template to be the region of the input image that the template was tracked to in $I_n(\mathbf{x})$:

**Strategy 2: Naive Update**

$$T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n)) \text{ for all } n \geq 1.$$

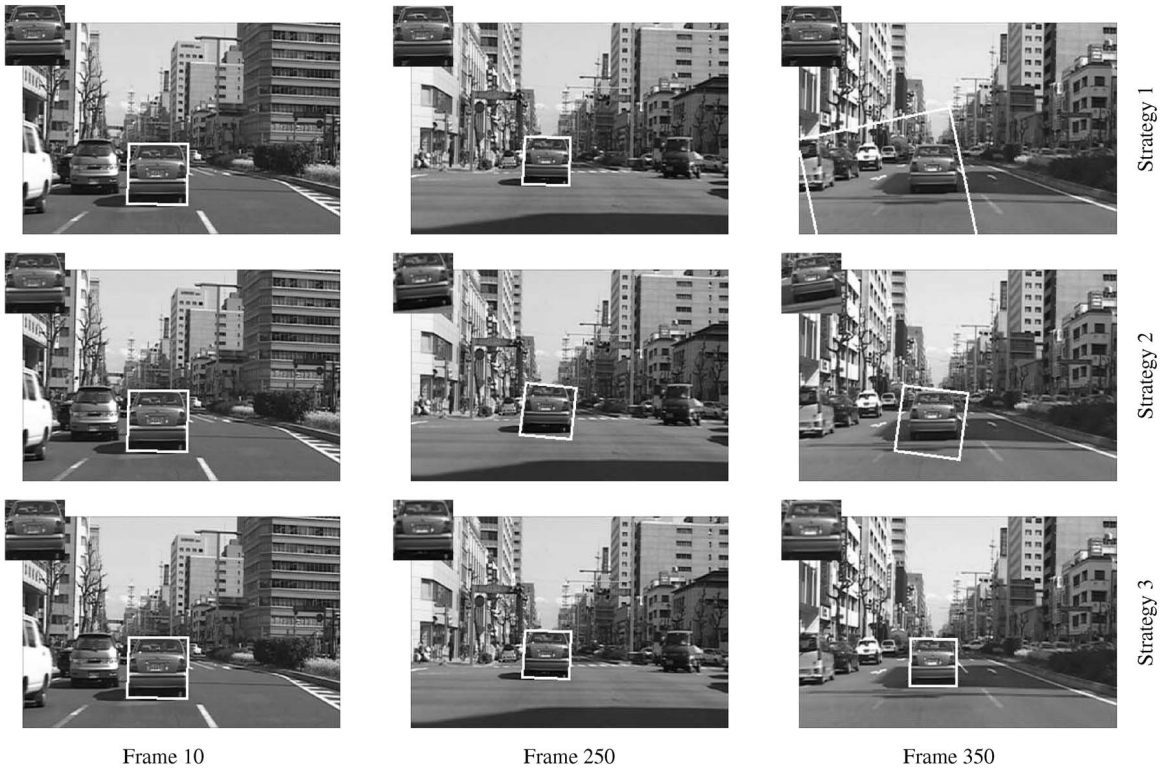


Fig. 1. A qualitative comparison of update Strategies 1, 2, and 3. With Strategy 1, the template is not updated and tracking eventually fails. With Strategy 2, the template is updated every frame and the template "drifts." With Strategy 3, the template is updated every frame, but a "drift correction" step is added. With this strategy, the object is tracked correctly and the template updated appropriately across the entire sequence.

Neither of these strategies are very good. With the first strategy, the template eventually, and inevitably, becomes out-of-date and no longer representative of the appearance of the object being tracked. With the second strategy, the template eventually drifts away from the object. Small errors in the warp parameters $\mathbf{p}_n$ mean that the new template $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$ is always a slighted shifted version of what it ideally should be. These errors accumulate and, after a while, the template drifts away from the object that it was initialized to track. See Fig. 1 for an example of the template drifting in this way. Note that simple variants of this strategy such as updating the template every few frames, although more robust, also eventually suffer from the same drifting problem.

How can we update the template every frame and avoid it wandering off? One possibility is to keep the first template $T_1(\mathbf{x})$ around and use it to correct the drift in $T_{n+1}(\mathbf{x})$. For example, we could take the estimate of $T_{n+1}(\mathbf{x})$ computed in Strategy 2 and then align $T_{n+1}(\mathbf{x})$ to $T_1(\mathbf{x})$ to eliminate the drift. Since $T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$, this is the same as first tracking in image $I_n(\mathbf{x})$ with template $T_n(\mathbf{x})$ and then with template $T_1(\mathbf{x})$. If the nonlinear minimizations in (2) and (3) are solved perfectly, this is theoretically exactly the same as just tracking with $T_1(\mathbf{x})$. The nonlinear minimizations are solved using a gradient descent algorithm, however, and, so, this strategy is actually different. Let us change the notation slightly to emphasize the point that a gradient descent algorithm is used to solve (3). In particular, rewrite (3) as:

$$\mathbf{p}_n = \mathrm{gd}\min_{\mathrm{p}=\mathrm{p}_{n-1}} \sum_{\mathrm{x}\in T_n} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_n(\mathbf{x})]^2, \tag{4}$$

where $\mathrm{gd}\min_{\mathrm{p}_{n-1}}$ means "perform a gradient descent minimization" starting at $\mathbf{p} = \mathbf{p}_{n-1}$. To correct the drift in Strategy 2, we therefore propose to compute updated parameters:

$$\mathbf{p}_n^* = \mathrm{gd}\min_{\mathrm{p}=\mathrm{p}_n} \sum_{\mathrm{x}\in T_1} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_1(\mathbf{x})]^2. \tag{5}$$

Note that this is different from tracking with the constant template $T_n = T_1$ using:

$$\mathrm{gd}\min_{\mathrm{p}=\mathrm{p}_{n-1}} \sum_{\mathrm{x}\in T_1} [I_n(\mathbf{W}(\mathbf{x};\mathbf{p})) - T_1(\mathbf{x})]^2 \tag{6}$$

because the starting point of the gradient descent is different. It is $\mathbf{p}_n$ rather than $\mathbf{p}_{n-1}$. To correct the drift, we use $\mathbf{p}_n^*$ rather than $\mathbf{p}_n$ to form the template for the next image. In summary (see also Fig. 2), we update the template using:

**Strategy 3: Template Update with Drift Correction**

$$\text{If } \|\mathbf{p}_n^* - \mathbf{p}_n\| \le \epsilon, \text{ then } T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$$
$$\text{else } T_{n+1}(\mathbf{x}) = T_n(\mathbf{x}),$$

where $\epsilon > 0$ is a small threshold that enforces the requirement that the result of the second gradient descent does not diverge too far from the result of the first. If it does, there must be a problem and so we act conservatively by not updating the template in that step. A minor variant of Strategy 3 is to perform the drift-correcting alignment using the magnitudes of the gradients of the image and the template rather than the raw images to increase robustness to illumination variation.

### 2.2 Qualitative Comparison

We now present a *qualitative* comparison of the strategies. Although we only have room to include one set of results, these results are typical. A *quantitative* evaluation is included in Section 2.4.

We implemented each of the three update strategies above and ran them on a 972 frame video of a car being tracked using the 2D similarity transform in (1). Sample frames are shown in Fig. 1

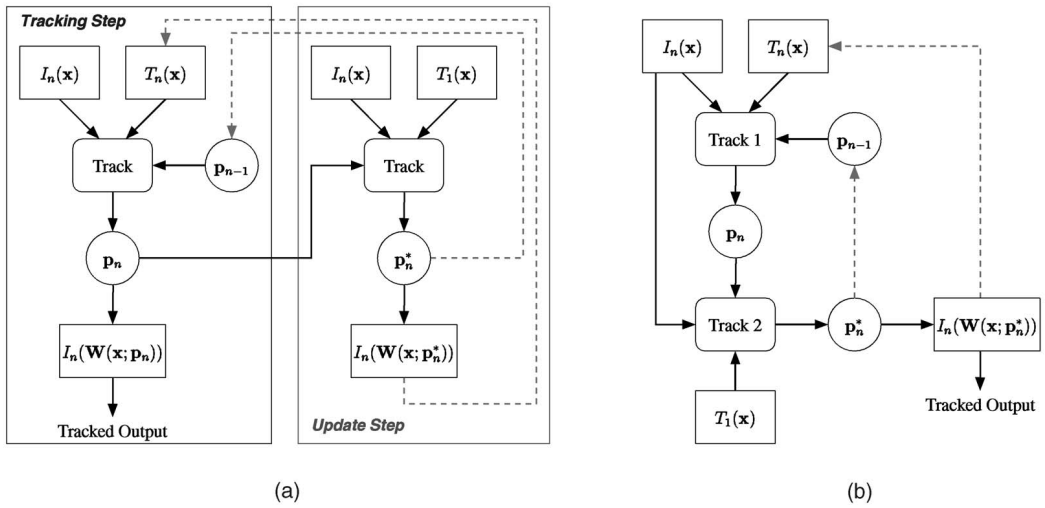


Fig. 2. (a) Update Strategy 3. (b) Update Strategy 3 (reorganized). Two equivalent schematic diagrams for update Strategy 3. The diagrams are equivalent in the sense that they result in exactly the same sequence of parameters $\mathbf{p}_n$. (a) Can be interpreted as first tracking with template $T_n$ and then updating $T_n$ using the drift correction step. (b) Can be interpreted as tracking with constant template $T_1$, after first tracking with $T_n(\mathbf{x}) = I_{n-1}(W(\mathbf{x};\mathbf{p}_{n-1}^*))$ to avoid local minima.

for each of the update algorithms. If the template is not updated (Strategy 1), the car is no longer tracked correctly after frame 312. If we update the template every frame using the naive approach (Strategy 2), by around frame 200 the template has drifted away from the car. With update Strategy 3 "Template Update with Drift Correction," the car is tracked throughout the entire sequence and the template is updated correctly in every frame, without introducing any drift. See the accompanying movie[1] "car-track.mpg" for tracking results on the sequence.

### 2.3 Reinterpretation of Update Strategy 3

A schematic diagram of Strategy 3 is included in Fig. 2a. The image $I_n(\mathbf{x})$ is first tracked (left box) with template $T_n(\mathbf{x})$ starting from the previous parameters $\mathbf{p}_{n-1}$. The result is the tracked image $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$ and the parameters $\mathbf{p}_n$. The new template $T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$ is then computed (right box) by tracking $T_1(\mathbf{x})$ in $I_n(\mathbf{x})$ starting at parameters $\mathbf{p}_n$.

If we reorganize Fig. 2a slightly, we get Fig. 2b. The only change made in this reorganization is that the "tracked output" is $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n^*))$ rather than $I_n(\mathbf{W}(\mathbf{x};\mathbf{p}_n))$. The difference between Figs. 2a and 2b is not the computation (the two diagrams result in the same sequence of parameters $\mathbf{p}_n$), but their interpretation. Fig. 2a can be interpreted as tracking with $T_n(\mathbf{x})$ followed by updating $T_n(\mathbf{x})$. Fig. 2b can be interpreted as tracking with $T_n(\mathbf{x})$ to get an initial estimate to track with $T_1(\mathbf{x})$. This initial estimate improves robustness because tracking with $T_1(\mathbf{x})$ is prone to local minima. Tracking with $I_{n-1}(\mathbf{W}(\mathbf{x};\mathbf{p}_{n-1}^*))$ is less prone to local minima and is used to initialize the tracking with $T_1(\mathbf{x})$ and start it close enough to avoid local minima. In summary, there are two equivalent ways to interpret Strategy 3:

1. The template can be updated every frame, but it must be realigned to the original template $T_1(\mathbf{x})$ to remove the drift that would otherwise build up.
2. Not updating the template and tracking using the constant template $T_1(\mathbf{x})$ is fine, so long as we first initialize $\mathbf{p}_n$ by tracking with $T_n(\mathbf{x}) = I_{n-1}(W(\mathbf{x};\mathbf{p}_{n-1}^*))$ to avoid local minima.

1. The movies can also be downloaded from http://www.ri.cmu.edu/projects/project_513.html.

### 2.4 Quantitative Evaluation

We now present a *quantitative* evaluation of Strategy 3 in the context of the second interpretation above. We measure how much more robust tracking is if we initialize it by first tracking with $I_{n-1}(W(\mathbf{x};\mathbf{p}_{n-1}^*))$; i.e., use Strategy 3 rather than Strategy 1.

Our goal is to track the car in the 972 frame video sequence shown in Fig. 1. First, using a combination of Lucas-Kanade tracking and hand reinitialization, we obtain a set of ground-truth parameters $\mathbf{p}_n$ for each frame. We then generate 50 test cases for each of the 972 frames by randomly perturbing the ground-truth parameters $\mathbf{p}_n$. Note that the ground truth is perturbed randomly for each frame, not just the first frame. In a sense, we evaluate the fitting robustness independently for each frame, and then average over the 972 images. The perturbation is computed using a normal distribution so that the root-mean-square template coordinate locations in the image are displaced by a known spatial standard deviation. We then run the two tracking algorithms starting with the same perturbed parameters and determine which of the two algorithms converged by comparing the final $\mathbf{p}_n$ with the ground-truth. A trial is said to have converged if all four corners of the template are within 1.0 pixels of the ground-truth locations. This experiment is repeated for all frames over a range of perturbation standard deviations. The final result is a graph plotting the frequency of convergence against the perturbation magnitude. The results of this comparison are shown in Fig. 3. We plot two curves, one for update Strategy 1 "No Update" and one for update Strategy 3 "Template Update with Drift Correction." No results are shown for Strategy 2 because, after a few frames, the template drifts and, so, none of the trials converge to the correct location. The accompanying movie "car-exp.mpg" shows example trials for both algorithms with the ground truth marked in yellow and the perturbed position tracked in green. Fig. 3 clearly demonstrates that updating the template using Strategy 3 dramatically improves the tracking robustness.

## 3 Template Tracking with Appearance Variation

We now consider the problem of template tracking with linear appearance variation. Instead of tracking with a single template $T_n(\mathbf{x})$ (for each frame $n$), we assume that a linear model of

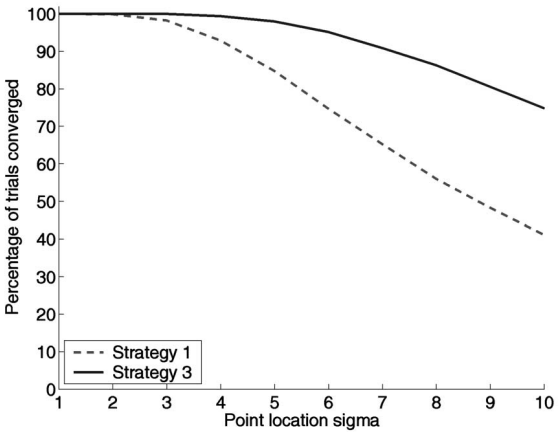


Fig. 3. The frequency of convergence of Strategies 1 and 3 plot against the magnitude of the perturbation to the ground-truth parameters, computed over 50 trials for each frame in the sequence used in Fig. 1. As can be seen, updating the template using Strategy 3 results in far more robust tracking.

appearance variation is used; i.e., a set of appearance images $A_n^i(\mathbf{x})$ where $i = 1, \ldots, d_n$. Instead of the template $T_n(\mathbf{x})$ appearing (appropriately warped) in the input image $I_n(\mathbf{x})$, we assume that:

$$T_n(\mathbf{x}) + \sum_{i=1}^{d_n} \lambda^i A_n^i(\mathbf{x}) \tag{7}$$

appears (appropriately warped) in the input image for a unknown set of *appearance parameters* $\boldsymbol{\lambda} = (\lambda^1, \ldots, \lambda^{d_n})^{\mathrm{T}}$. The appearance images $A_n^i(\mathbf{x})$ can be used to model either illumination variation [6] or more general linear appearance variation [4], [5]. In this paper, we focus particularly on Active Appearance Models [5], [8] which combine a linear appearance model with a (low parametric) piecewise affine warp to model the shape deformation $\mathbf{W}(\mathbf{x}; \mathbf{p})$. The process of tracking with such a linear appearance model then consists of minimizing:

$$(\mathbf{p}_n, \boldsymbol{\lambda}_n) = \arg\min_{(\mathrm{p}, \boldsymbol{\lambda})} \sum_{\mathbf{x} \in T_n} \left[ I_n(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T_n(\mathbf{x}) - \sum_{i=1}^{d_n} \lambda^i A_n^i(\mathbf{x}) \right]^2. \tag{8}$$

Several efficient gradient descent algorithms have been proposed to solve this nonlinear optimization problem including [6] for translations, affine warps, and 2D similarity transformations, [1] for arbitrary warps that form a group, and [8] for Active Appearance Models. Denote the result:

$$(\mathbf{p}_n, \boldsymbol{\lambda}_n) = \mathrm{gd} \min_{(\mathrm{p}_{n-1}, \boldsymbol{\lambda}_{n-1})} \sum_{\mathbf{x} \in T_n} \left[ I_n(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T_n(\mathbf{x}) - \sum_{i=1}^{d_n} \lambda^i A_n^i(\mathbf{x}) \right]^2, \tag{9}$$

where the gradient descent is started at $(\mathbf{p}_{n-1}, \boldsymbol{\lambda}_{n-1})$.

### 3.1 Updating both the Template and the Appearance Model

Assume that the initial template $T_1$ and appearance model $A_1^i$ are given. The template update problem with linear appearance variation consists of estimating $T_{n+1}$ and $A_{n+1}^i$ from $I_0(\mathbf{x}), \ldots, I_n(\mathbf{x})$, $T_1(\mathbf{x}), \ldots, T_n(\mathbf{x})$, and $A_1^i, \ldots, A_n^i$. Analogously to the above, denote the result of aligning with respect to the initial template $T_1$ and appearance model $A_1^i$, but starting the gradient descent from the result of aligning with respect to the current template $T_n$ and appearance model $A_n^i$, as follows:

$$(\mathbf{p}_n^*, \boldsymbol{\lambda}_n^*) = \mathrm{gd} \min_{(\mathrm{p}_n, \boldsymbol{\lambda}_n)} \sum_{\mathbf{x} \in T_n} \left[ I_n(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T_1(\mathbf{x}) - \sum_{i=1}^{d_1} \lambda^i A_1^i(\mathbf{x}) \right]^2. \tag{10}$$

One way to update the template and appearance model is then as follows:

**Strategy 4: Template and Appearance Model Update with Drift Correction**

$$\begin{aligned} &\text{If } \|\mathbf{p}_n^* - \mathbf{p}_n\| \leq \epsilon, \\ &\text{then } (T_{n+1}(\mathbf{x}), A_{n+1}^i) = \mathrm{PCA}(I_1(\mathbf{W}(\mathbf{x}; \mathbf{p}_1^*)), \ldots, I_n(\mathbf{W}(\mathbf{x}; \mathbf{p}_n^*))) \\ &\text{else } T_{n+1}(\mathbf{x}) = T_n(\mathbf{x}), A_{n+1}^i = A_n^i, \end{aligned}$$

where PCA() means perform Principal Components Analysis setting $T_n$ to be the mean and $A_n^i$ to be the first $d_n$ eigenvectors, where $d_n$ is chosen to keep a fixed amount of the energy, typically 95 percent. (Other variants of this exist, such as incrementally updating appearance model $A_n^i$ to include the new measurement $I_n(W(\mathbf{x}; \mathbf{p}_n^*))$.) If we reinterpret this algorithm as in Section 2.3, we end up with the following two step tracking algorithm:

**Step 1**: Apply PCA to $I_1(\mathbf{W}(\mathbf{x}; \mathbf{p}_1^*)), \ldots, I_{n-1}(W(\mathbf{x}; \mathbf{p}_{n-1}^*))$. Set $T_n$ to be the mean vector and $A_n^i$ to be the first $i = 1, \ldots, d_n$ eigenvectors. Once computed, track with template $T_n$ and appearance model $A_n^i$.

**Step 2**: Track with the a priori template $T_1(\mathbf{x})$ and linear appearance model $A_1^i(\mathbf{x})$, starting the gradient descent at the result of the first step.

One way to interpret these two steps is as performing "progressive appearance complexity," analogously to "progressive transformation complexity" [3], the standard heuristic for improving the robustness of tracking algorithms by increasing the complexity of the warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$. For example, tracking with an affine warp is often performed by first tracking with a translation, then a 2D similarity transformation and, finally, a full affine warp. Here, tracking with one appearance model is used to initialize tracking with another. Based on this analogy, we add another step to the algorithm above:

**Step 0**: Track using the template $T_n(\mathbf{x}) = I_{n-1}(W(\mathbf{x}; \mathbf{p}_{n-1}^*))$ with *no* appearance model.

This step is performed before the two steps above and is used to initialize them.

### 3.2 Quantitative Evaluation

We evaluate Strategy 4 "Template and Appearance Model Update with Drift Correction" in the same way that we evaluated Strategy 3 in Section 2.4. We use a 600 frame video of a face and construct an initial AAM for it by hand-marking feature points in a random selection of 80 frames. We then generate ground-truth parameters by tracking the AAM through the video using a combination of AAM fitting [8], pyramid search, progressive transformation complexity, and reinitialization by hand. The accompanying movie "face-gt.mpg" plots the ground-truth AAM feature points on all images in the video sequence. The sequence shows the face of a car driver and includes moderate face pose and lighting variation. We generate 50 test cases for each of the 600 frames in the video by randomly perturbing the AAM parameters. Similarly to Section 2.4 and following the exact procedure in [8], we generate perturbations in both the similarity transform of the AAM and the shape parameters. Specifically, the RMS similarity displacement standard deviation is chosen to be four times the shape eigenvector standard deviation so that each is weighted

according to their relative importance. For each test case, we compared four algorithms:

**Algorithm 1**: Step 2 (no update).
**Algorithm 2**: Step 1 followed by Step 2.
**Algorithm 3**: Step 0 followed by Step 1 followed by Step 2.
**Algorithm 4**: Step 0 followed by Step 2.

We plot the frequency of convergence of these four algorithms computed on average across all $50 \times 600$ test cases against the magnitude of the perturbation to the AAM parameters in Fig. 4. (The accompanying movie "`face-exp.mpg`" shows example trials for one of the algorithms with the ground truth marked in yellow and the perturbed position tracked in green.) As for the single template tracking case in Section 2, the template and appearance model update algorithms (Algorithms 2, 3, and 4) all outperform the algorithm which does not update the template and appearance mode (Algorithm 1). As one might imagine, Algorithm 3 (Steps 0, 1, and 2) marginally outperforms Algorithm 2 which just uses Steps 1 and 2. Algorithm 4 performs significantly worse than both Algorithms 2 and 3 indicating that Step 1 is essential for the best performance. Finally, we also plot a curve for a fifth algorithm. Algorithm 5 consists of tracking the sequence with the final AAM computed by the update algorithm; i.e., we use constant template $T_{600}(\mathbf{x})$ and constant linear appearance model

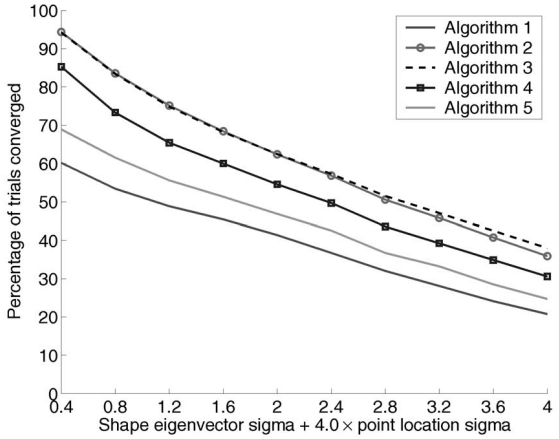


Fig. 4. A comparison of the frequency of convergence of five template and appearance model update algorithms. The three algorithms which actually update the template and/or appearance model (Algorithms 2, 3, and 4) all dramatically outperform the algorithms which do not (Algorithms 1 and 5).

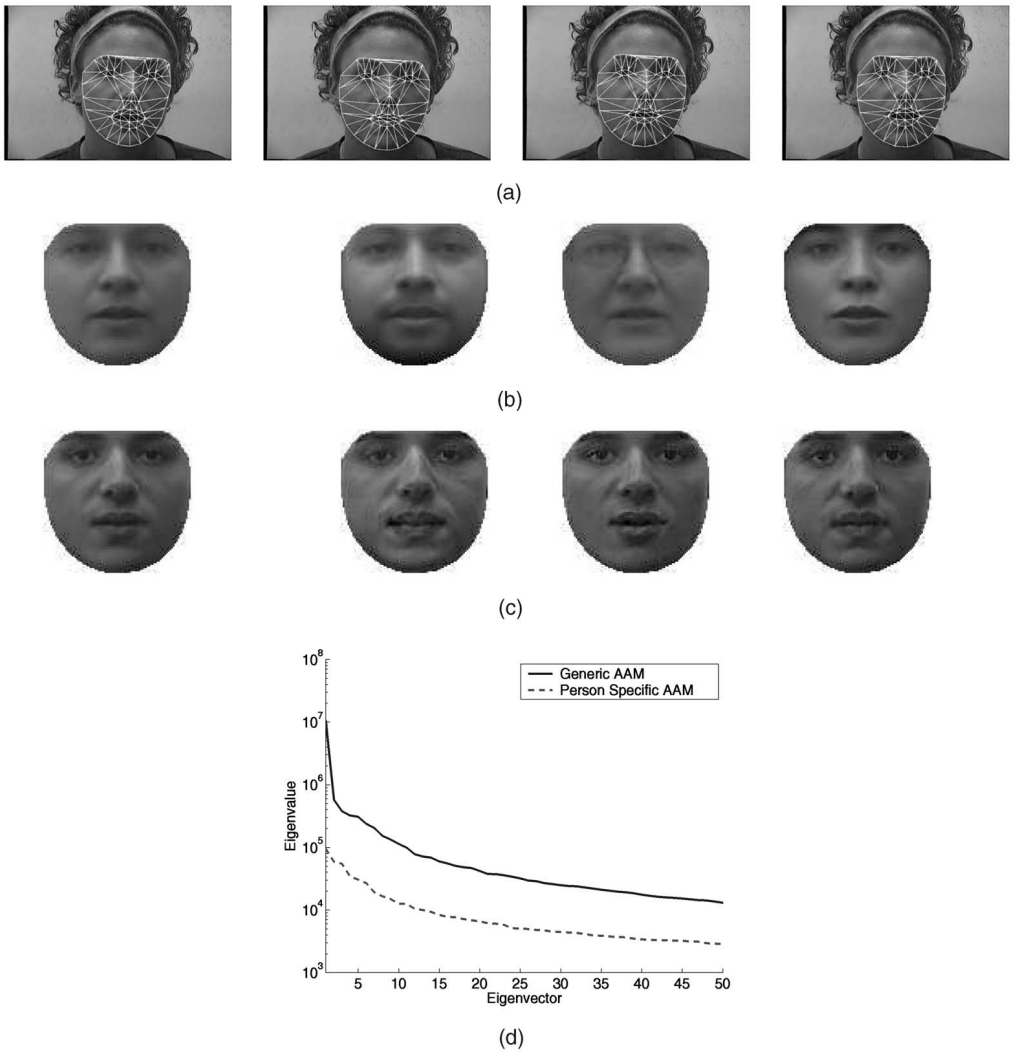


Fig. 5. An illustration of the conversion of a generic AAM to a person-specific AAM. (a) Four frames from the video that is tracked. (b) An illustration of the appearance variation of the generic AAM. (c) An illustration of the appearance variation of the computed person-specific AAM. (d) The appearance eigenvalues of the two AAMs. Note how the person-specific AAM requires far fewer parameters and just codes illumination variation, whereas the generic AAM mainly codes identity variation.

$A^i_{600}(\mathbf{x})$. Since this is an AAM computed using all 600 frames in the sequence, the performance is significantly better than the a priori AAM computed using only 80 frames. The results are still not as good as Algorithm 2 where the AAM is updated every frame.

### 3.3 Converting a Generic AAM into a Person-Specific AAM

When we use Step 1 above, a new template and appearance model are computed online as we track the face through the video. To illustrate this process, we applied Algorithm 2 to track a video of a face using a generic, person-independent AAM. The accompanying movie "`face-app.mpg`" shows the tracked face, $T_1(\mathbf{x})$ and the first two $A^i_1(\mathbf{x})$. Also, shown are the current $T_n(\mathbf{x})$ and the first two $A^i_n(\mathbf{x})$ for each frame. The result is that at the end of the sequence, the template and appearance model update algorithm has computed a person-specific appearance model.

This process is illustrated in Fig. 5. Fig. 5a shows four frames of the face that is tracked. Note that no images of the person in the video were used to compute the generic AAM. Fig. 5b shows the appearance eigenvectors of the generic AAM. Note that the appearance eigenvectors mainly code identity variation. Fig. 5c shows the appearance eigenvectors of the person-specific AAM computed using our algorithm. Note that the eigenvectors mainly code illumination variation, and no identity variation. Fig. 5d plots the appearance eigenvalues of both AAMs. There is far less appearance variation in the person-specific AAM and it therefore requires far fewer appearance parameters to provide the same representational power.

## 4 Conclusion

We have investigated the template update problem. We first proposed a template update algorithm that does not suffer from the "drift" inherent in the naive algorithm. Next, we showed how this algorithm can be reinterpreted as a heuristic to avoid local minima and quantitatively evaluated it as such. The results show that updating the template using "Template Update with Drift Correction" improves tracking robustness. We then extended our algorithm to template tracking with linear appearance models and quantitatively compared five variants of the update strategy. The results again show that updating both the template and the appearance model with drift correction results in more robust fitting. Finally, we showed that our linear appearance model update strategy can also automatically compute a person-specific AAM while tracking with a generic AAM. Note that updating a template can be regarded as one form of *unsupervised model building*. As such, it is related to the growing body of work on that problem, one example of which is [9].

One implication of the success of our algorithm is that it shows that when the appearance of an object changes, the result is to make the template tracking problem more susceptible to local minima. Informally, the local minima must get "closer" or the global minima (i.e., more correctly, the one corresponding to correct tracking) "less deep." If the template is not updated, the tracking algorithm eventually falls into one of the local minima and tracking fails. A template extracted from the previous frame is far less likely to suffer from these local minima than the original template because the appearance variation is less.

Our algorithm suffers from a number of limitations. First, updating the template (and the appearance model) dramatically increases the computational cost of tracking because much of the cost of tracking only needs to be performed once per template (and appearance model) [6], [2], [8]. Although, in our experimental results, the template is updated every frame, this is just to illustrate the drift problem more clearly. Instead of updating the template every frame, it could just be updated whenever it is determined that it has changed significantly, thereby reducing the average computational cost.

Second, our algorithm only covers the case where the visibility of the object being tracked does not change. We have not attempted to address the question of how to update a template when new parts of the object come into view. For example, when tracking a human head with a cylinder model, different parts of the head come into view as the head rotates [10]. Our algorithm will not help with such scenarios. We have concentrated on the case that the visibility is constant, but the appearance changes. Extending our algorithm to combine it with techniques for extending the template when the visibility changes [10] is left as future work.

## Acknowledgments

The research described in this report was partially supported by Denso Corporation, Japan, and was conducted at Carnegie Mellon University while Takahiro Ishikawa was a visiting industrial scholar. This research was also supported, in part, by the US Department of Defense through award number N41756-03-C4024. The generic AAM model in Section 3.3 was trained on the ViaVoice™ AV database provided by IBM Research. The authors also thank Bob Collins and the anonymous reviewers.

## References

[1] S. Baker and I. Matthews, "Equivalence and Efficiency of Image Alignment Algorithms," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 1090-1097, 2001.

[2] S. Baker and I. Matthews, "Lucas-Kanade 20 Years on: A Unifying Framework," *Int'l J. Computer Vision*, vol. 53, no. 3, pp. 221-255, 2004.

[3] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, "Hierarchical Model-Based Motion Estimation," *Proc. European Conf. Computer Vision*, pp. 237-252, 1992.

[4] M. Black and A. Jepson, "Eigen-Tracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Int'l J. Computer Vision*, vol. 36, no. 2, pp. 63-84, 1998.

[5] T. Cootes, G. Edwards, and C. Taylor, "Active Appearance Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681-685, June 2001.

[6] G. Hager and P. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025-1039, Oct. 1998.

[7] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. Int'l Joint Conf. Artificial Intelligence*, pp. 674-679, 1981.

[8] I. Matthews and S. Baker, "Active Appearance Models Revisited," *Int'l J. Computer Vision*, 2004, also appeared as Technical Report CMU-RI-TR-03-02, Robotics Inst., Carnegie Mellon Univ., Pittsburgh, Penn.

[9] T. Vetter, M. Jones, and T. Poggio, "A Bootstrapping Algorithm for Learning Linear Models of Object Classes," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 40-46, 1997.

[10] J. Xiao, T. Kanade, and J. Cohn, "Robust Full-Motion Recovery of Head by Dynamic Templates and Re-Registration Techniques," *Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition*, pp. 163-169, 2002.